Poster Paper in Proceedings of the 16th Annual NACCQ, Palmerston North, New Zealand July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz

Complexity: A vital Aspect of Learning Programming

John McPhee

Christchurch Polytechnic Institute of Technology
Christchurch, New Zealand
mcpheej@cpit.ac.nz

This poster reviews research on software complexity. Cant et al (1995) defined complexity as the characteristics of software which affect the level of resources used by a person performing a given task on it. Complexity is a concept of paramount importance to computing. It is central to most facets of software development.

Software development effort estimation methods include a complexity factor in determining the development time required. Given this contributory role, improving our understanding of the nature of complexity offers the potential to help speed up the software development process. It may also improve our ability to effectively teach programming.

Considerable research has been carried out in an attempt to devise an effective measure of complexity. The results of this research are not totally convincing, with no clearly identified metric emerging as yet. McCabe's cyclomatic complexity and Halstead's program volume are two of the best known metrics. More recent research has broadened by taking into account the programmer, and redefining complexity as an interaction between aspects of the software code and the programmer. Early research in this area has been done by Ehrlich & Soloway with their 'control flow plans' and 'variable plans', builds on the concept of 'chunking'. This wider perspective has provided room to incorporate comprehension and cognitive styles as contributory factors to software complexity; concepts that have high face validity yet have been given little attention to date.

Combining and reapplying the research from these two approaches, promises to provide a much stronger understanding of what constitutes program complexity and thus help identify significantly more accurate estimates of development effort, optimal programming languages, and improved programming teaching methods.

References

Cant, Jeffrey, & Henderson-Sellers (1995) " A Conceptual Model of Cognitive Complexity of Elements of the programming process". Information and Software Technology 1995 37(7)

Ehrlich,K. & Soloway,E.(1984) "An empirical investigation of the tacit plan knowledge in programming" in Thomas, J.C. and Schneider,M.L. (eds) Human Factors in Computer Systems Ablex Publishing (1984) pp113 - 133

Halstead (1987) Elements of Software Science, Elsevier/North Holland

McCabe & Butler (1989) "Design Complexity Measurement and Testing" Communications of the ACM, Dec 1989, 32(12)